A9

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/845,693	04/30/2001	Erik R. Altman	Y0R9-2000-0844 US (8728-4	2678

46069 7590 08/08/2006

F. CHAU & ASSOCIATES, LLC
130 WOODBURY ROAD
WOODBURY, NY 11797

EXAMINER	
HUISMAN, DAVID J	

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 08/08/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	09/845,693	ALTMAN ET AL.
	Examiner	Art Unit
	David J. Huisman	2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.
- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *25 May 2006*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-3,5-13 and 15-21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-3,5-13 and 15-21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *30 April 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-3, 5-13, and 15-21 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: Amendment as received on 5/25/2006.

Specification

3. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1-11, 13, and 15-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lavi et al., U.S. Patent No. 6,453,407 (as applied in the previous Office Action and herein referred to as Lavi), in view of Ito et al., U.S. Patent No. 5,742,782 (herein referred to as Ito).

6. Referring to claim 1, Lavi has taught a method for processing a first instruction set and a second instruction set in a processor comprising the steps of:

a) providing a program of instructions comprising a plurality of instructions of the first instruction set and a plurality of instructions of the second instruction set, wherein the plurality of instructions of the first instruction set are decoded by a decoder in an execution pipeline and the plurality of instructions of the second instruction set are predecoded by a compiler. See the abstract and Fig.3 and Fig.6, for instance. The first set of instructions (set of non-predecoded instructions) comes from program memory and is decoded by decoder 40. The second set of instructions (set of predecoded instructions) is stored in predecoded form in CLIW array 70 by a compiler.

b) storing the plurality of instructions of the second set in a plurality of buffers proximate to a plurality of execution units. See Fig.3, component 70, and Fig.4, and note that predecoded instructions are stored in a plurality of buffers which are collectively referred to as a CLIW array.

c) executing at least one instruction of the first instruction set in response to a first counter. First set instructions are fetched from program memory based on a program counter (PC), which is inherently present. The PC is used to hold the current location from which to fetch an instruction. And, consequently, the next instruction address can also be determined from it.

d) executing at least one instruction of the second instruction set in response to at least a second counter, wherein the second counter is invoked by a branch instruction of the first instruction set. When a branch instruction of the first set is decoded, there will be a pointer to a location in the CLIW array (see Fig.6). Hence this address that the decode outputs is a second program counter (recall that a PC is simply a register for holding the current instruction address).

e) Lavi has not explicitly taught that the step of executing at least one instruction of the second instruction set further comprises the steps of de-gating a plurality of execution queues storing the plurality of instructions of the first instruction set and pausing a fetching of the first instruction set from a memory. However, Ito has taught that an instruction decoder includes an instruction buffer (execution queue) for queuing instructions until they are dispatched. See Fig.2, component 31. In Ito, instructions will wait in the buffer until a plurality of gates (making up the rest of the logic in the decode in Ito) send an issue signal on line 133 which activates the dispatcher 36. Buffers/queues are useful for holding instructions until the conditions are such that the instruction can continue to progress through the pipeline. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Lavi to include an execution queue to store instructions of the first set. Now, looking at the abstract of Lavi, it is disclosed that an instruction of the first set is replaced by at least one predecoded instruction. Therefore, instead of the system issuing the instruction of the first set, at least one instruction of the second set is issued. In order to not issue this instruction of the first set, the gates of Ito would need to be controlled such that the instruction would be maintained in the execution queue. So, with the combination of Lavi and Ito, when a CLIW instruction (of the second set) is to issue, the first set instruction will be paused from issuing by having gates control the execution queue in which the instruction is stored. When a CLIW instruction is not to issue, then the first set instruction will be dispatched from the queue normally.

7. Referring to claim 2, Lavi in view of Ito has taught a method as described in claim 1. Lavi has further taught that instructions of the first set and instructions of the second set are

generated by a compiler and that instructions of the second set are statically loaded into the plurality of buffers as control signals ready for execution. See the abstract and Fig.3.

8. Referring to claim 3, Lavi in view of Ito has taught a method as described in claim 2. Furthermore, although not explicitly stated, it is possible in Lavi that instructions of the second set are more frequently executed that instructions of the first set. Fort instance, looking at Fig.6, the program could easily contain more reference instructions than regular instructions, which would cause more frequent execution of second set instructions. Fig.8 shows an example of this. Also, according to the abstract, the more that second set instructions are executed, the more power that is saved because less decoding is performed. Therefore, it would be obvious to gear programs towards executing second set instructions.

9. Referring to claim 5, Lavi in view of Ito has taught a method as described in claim 1. Lavi has further taught that the step of executing at least one instruction of the second instruction set further comprises the steps of fetching at least one instruction of the second instruction set from a buffer of the plurality of buffers and sequencing the at least one instruction of the second instruction set to the execution units. See Fig.3 and Fig.6.

10. Referring to claim 6, Lavi in view of Ito has taught a method as described in claim 1. Lavi has further taught that the second instruction set is a logical subset of the first instruction set. See Fig.6, and note that the referenced instructions are predecoded versions of the regular instructions.

11. Referring to claim 7, Lavi in view of Ito has taught a method as described in claim 1. Lavi has further taught that the step of executing at least one instruction of the first instruction set further comprises the steps of fetching an instruction of the first set from a memory, decoding

the instruction, and issuing the decoded instruction to at least one execution unit. Se Fig.3 and note first set instructions are fetched from program memory, decoded, and then passed to computation units.

12. Referring to claim 8, Lavi in view of Ito has taught a method as described in claim 1. Lavi has not explicitly taught that a return to fetching of the first instruction set is signaled by a switch bit in a buffer of a branch unit storing instructions of the second instruction set. However, it should be realized that if an instruction in the CLIW array is a branch and the system branches to an address of an instruction that is not in the CLIW array, then the first set instruction will be fetched and executed. The branch instruction comprises a sequence of opcode bits, and all of the opcode bits define the branch instruction. Any one of these opcode bits may be considered the switch bit because it plays a part in causing the branch to occur.

13. Referring to claim 9, Lavi in view of Ito has taught a method as described in claim 1. Lavi has not explicitly taught that a return to fetching of the first instruction set is signaled by a return instruction of the second instruction set stored in a buffer of a branch unit. However, it should be realized that if an instruction in the CLIW array is a branch and the system branches to an address of an instruction that is not in the CLIW array, then the first set instruction will be fetched and executed. This is equivalent to a return instruction, and Lavi has not ruled out that CLIW instructions can be branches/returns.

14. Referring to claim 10, Lavi in view of Ito has taught a method as described in claim 1. Lavi has further taught that each execution unit is associated with a different buffer of the plurality of buffers. It should be realized that each buffer holds a different type of instruction. That instruction will correspond to a particular unit (and consequently, that buffer will

correspond to that particular unit). For instance, if a buffer is folding a floating-point arithmetic instruction, the buffer will correspond to the floating-point arithmetic unit and not to the integer unit or branch unit.

15. Referring to claim 11, Lavi has taught a processor for processing a program of instructions comprising instructions of a first instruction form and a second instruction form comprising:

a) a plurality of execution units for receiving instructions. See Fig.3, component 60 (note the multiple computation units).

b) a branch unit (see column 3, lines 29-34, and note that branches may be executed, and consequently, a branch unit exists) connected to an instruction fetch unit (see Fig.3, component 22, for instance) for the first instruction form and a sequencer (see column 10, lines 1-7) for the second instruction form.

c) a decode unit for decoding instructions of the first instruction form into control signals for the execution units. See Fig.3, component 40.

d) a plurality of buffers, proximate to the execution units, for storing predecoded instructions of the second instruction form. See Fig.3, component 70, and Fig.4, and note that predecoded instructions are stored in a plurality of buffers which are collectively referred to as a CLIW array.

e) Lavi has not explicitly taught wherein the sequencer controls a plurality of gates connected between a plurality of execution queues for storing decoded instructions of the first instruction form and the plurality of execution units. However, Ito has taught that an instruction decoder includes an instruction buffer (execution queue) for queuing instructions until they are dispatched. See Fig.2, component 31. In Ito, instructions will wait in the buffer until a plurality of gates (making up the rest of the logic in the decode in Ito) send an issue signal on line 133 which activates the dispatcher 36. Buffers/queues are useful for holding instructions until the conditions are such that the instruction can continue to progress through the pipeline. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Lavi to include an execution queue to store instructions of the first form. Now, looking at the abstract of Lavi, it is disclosed that an instruction of the first form is replaced by at least one predecoded instruction. Therefore, instead of the system issuing the instruction of the first form, at least one instruction of the second form is issued. In order to not issue this instruction of the first form, the gates of Ito would need to be controlled such that the instruction would be maintained in the execution queue. So, with the combination of Lavi and Ito, when a CLIW instruction (second form) is to issue, the first form instruction will be prevented from issuing by having gates control the execution queue in which the instruction is stored. When a CLIW instruction is not to issue, then the first form instruction will be dispatched from the queue normally.

16. Referring to claim 13, Lavi in view of Ito has taught a processor as described in claim 11. Lavi has further taught that the sequencer, engaged by the branch unit, addresses the predecoded instructions of the second instruction form stored in the buffers and sequences predecoded instructions of the second instruction form to the execution units. See Fig.6, for instance, and note that a first form instruction acts as a branch (i.e., when encountered, it tells the system to branch to an address in the CLIW array). Then the fetching begins from that part of the array.

17. Referring to claim 15, Lavi in view of Ito has taught a processor as described in claim 11. Lavi has further explicitly taught that each execution unit is associated with a different buffer of the plurality of buffers. It should be realized that each buffer holds a different type of instruction. That instruction will correspond to a particular unit (and consequently, that buffer will correspond to that particular unit). For instance, if a buffer is folding a floating-point arithmetic instruction, the buffer will correspond to the floating-point arithmetic unit and not to the integer unit or branch unit.

18. Referring to claim 16, Lavi in view of Ito has taught a processor as described in claim 11. Lavi has further taught that the branch unit switches the processor from the first instruction form to the second instruction form in response to a branch instruction of the first instruction form. Again, looking at Fig.6, it can be seen that a first form instruction acts as a branch (i.e., when encountered, it tells the system to branch to an address in the CLIW array). Then the fetching begins from that part of the array.

19. Referring to claim 17, Lavi in view of Ito has taught a processor as described in claim 11. While Lavi has not explicitly taught that the branch unit switches the processor from the second instruction form to the first instruction form in response to a branch instruction of the second instruction form, it should be realized that if an instruction in the CLIW array is a branch and the system branches to an address of an instruction that is not in the CLIW array, then the first form instruction will be fetched and executed.

20. Referring to claim 18, Lavi in view of Ito has taught a processor as described in claim 11. Lavi has not explicitly taught a switch bit in a buffer of the plurality of buffers connected to the branch unit signals the sequencer to stop fetching from the buffers and enables instruction

fetching from a memory storing instructions of the first instruction form. However, it should be realized that if an instruction in the CLIW array is a branch and the system branches to an address of an instruction that is not in the CLIW array, then the first form instruction will be fetched and executed. The branch instruction comprises a sequence of opcode bits, and all of the opcode bits define the branch instruction. Any one of these opcode bits may be considered the switch bit because it plays a part in causing the branch to occur.

21. Referring to claim 19, Lavi in view of Ito has taught a processor as described in claim 11. While Lavi has not explicitly taught that an execution bandwidth of the execution units is larger than a fetch/issue bandwidth of the first form, looking at Fig.3, it can be seen that there are at least 4 execution units for executing up to 4 instructions per cycle. Official Notice is taken that dependency hazards and stalling in response to those hazards is well known in the art. In a pipelined processor, it is a common situation that the maximum bandwidth cannot be utilized because an instruction must be stalled due to some dependency. As a result, due to common pipeline stalling situations, it would have been obvious for the fetch/issue bandwidth of the first form to be less than the execution bandwidth.

22. Referring to claim 20, Lavi in view of Ito has taught a processor as described in claim 11. Lavi has further taught that the second instruction form is be a logical subset of the first instruction form, wherein the predecoded instructions of the second instruction form are statically stored in the plurality of buffers, and wherein the predecoded instructions of the second instruction form are control signals generated by a compiler and are not decoded during a runtime of the program. However, Lavi has taught such a concept. See column 11, lines 25-59, and note that instructions may be predecoded by the compiler and stored in an array of buffers so

that runtime decoding for those instructions may be avoided. The second form is also a subset of the first form. See Fig.6, and note that only some of the second form instructions correspond to first form instructions.

23. Referring to claim 21, Lavi has taught a processor for processing a first instruction form and a second instruction form of an instruction set comprising:

a) a plurality of execution units for receiving instructions. See Fig.3, component 60 (note the multiple computation units).

b) a branch unit connected to an instruction fetch unit for the first instruction form and a sequencer for the second instruction form, wherein the branch unit switches the processor from the first instruction form to the second instruction form in response to a branch instruction of the first instruction form. See Fig.6, for instance, and note that a first form instruction acts as a branch (i.e., when encountered, it tells the system to branch to an address in the CLIW array). Then the fetching begins from that part of the array.

c) while Lavi has not explicitly taught switching the processor from the second instruction form to the first instruction form in response to a branch instruction of the second instruction form, it should be realized that if an instruction in the CLIW array is a branch and the system branches to an address of an instruction that is not in the CLIW array, then the first form instruction will be fetched and executed.

c) a decode unit adapted to decode instructions of the first instruction form into control signals for the execution units. See Fig.3, component 40.

d) an issue unit adapted to sequence decoded instructions of the first instruction form. Fig.3, component 40 shows control signals being sent to the execution units. This is issuing the instruction.

e) a plurality of buffers, proximate to the execution units, for statically storing predecoded instructions of the second instruction form. See Fig.3, component 70, and Fig.4, and note that predecoded instructions are stored in a plurality of buffers which are collectively referred to as a CLIW array. These instructions are stored statically by a compiler.

f) each execution unit is connected to a corresponding buffer of the plurality of buffers. It should be realized that each buffer holds a different type of instruction. That instruction will correspond to a particular unit (and consequently, that buffer will correspond to that particular unit). For instance, if a buffer is folding a floating-point arithmetic instruction, the buffer will correspond to the floating-point arithmetic unit and not to the integer unit or branch unit.

g) Lavi has not taught that the sequencer, engaged by the branch unit, adapted to fetch the predecoded instructions and sequence the predecoded instructions of the second instruction form, wherein the sequencer is connected to a plurality of gates connected between a plurality of execution queues adapted to store the decoded instructions of the first instruction form and the plurality of execution units, the sequencer further adapted to control the gates. However, Ito has taught that an instruction decoder includes an instruction buffer (execution queue) for queuing instructions until they are dispatched. See Fig.2, component 31. In Ito, instructions will wait in the buffer until a plurality of gates (making up the rest of the logic in the decode in Ito) send an issue signal on line 133 which activates the dispatcher 36. Buffers/queues are useful for holding instructions until the conditions are such that the instruction can continue to progress through the pipeline. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Lavi to include an execution queue to store instructions of the first form. Now, looking at the abstract of Lavi, it is disclosed that an instruction of the first form is replaced by at least one predecoded instruction. Therefore, instead of the system issuing the instruction of the first form, at least one instruction of the second form is issued. In order to not issue this instruction of the first form, the gates of Ito would need to be controlled such that the instruction would be maintained in the execution queue. So, with the combination of Lavi and Ito, when a CLIW instruction (second form) is to issue, the first form instruction will be prevented from issuing by having gates control the execution queue in which the instruction is stored. When a CLIW instruction is not to issue, then the first form instruction will be dispatched from the queue normally.

24. Claim 12 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lavi in view of Ito, as applied above, and further in view of Ball and Larus, "Efficient Path Profiling," 1996 (as applied in the previous Office Action and herein referred to as Ball).

25. Referring to claim 12, Lavi in view of Ito has taught a processor as described in claim 11.

a) Lavi in view of Ito has not taught that the instructions of the first form and instructions of the second form are generated based on execution frequency, wherein instructions of the second form are executed more frequently than instructions of the first form. However, Ball has taught that measuring execution frequency is used to guide compilation and optimization. Lavi has stated in the abstract, that storing instructions in the CLIW array prevents decoding of these instructions, which consequently saves power. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Lavi to include the compilation techniques of Ball so that Lavi would be able to determine which instructions are most frequently executed. Once that knowledge is known, the most frequently executed instructions would be the ones stored in the CLIW array. This would result in less overall decoding during run-time (since the most frequently executed instructions are already predecoded and stored in the array), thereby saving the most amount of power.

Response to Arguments

26. Applicant's arguments filed on May 25, 2006, have been fully considered but they are not persuasive.

27. Applicant argues the novelty/rejection of claim 1 on page 10 of the remarks, in substance that:

> "Lavi teaches a single instruction set stored as decoded and undecoded instructions. Decoded and undecoded instructions of the same instruction set are not analogous to a first instruction set and a second instruction set, essentially as claimed in claim 1."

28. These arguments are not found persuasive for the following reasons:

a) First and second instruction sets are broadly interpreted as first and second sets of instructions. Any group of instructions may be considered a set of instructions. In addition, the abstract states that the first set of instructions is a regular set while the second set is a CLIW set.

29. Applicant argues the novelty/rejection of claim 11 on page 11 of the remarks, in substance that:

> "Lavi teaches that information that will be read from the CLIW array is fetched according to a CLIW reference instruction - the CLIW reference instruction is fetched from the same pathway as regular instructions. Thus, Lavi fails to teach or suggest de-gating a plurality of execution queues

> storing the plurality of instructions of the first instruction form. The normal instructions and the reference instructions are fetched along the same pathway (see col. 5, lines 13-21) - thus de-gating the normal instructions would also de-gate the reference instructions."

30. These arguments are not found persuasive for the following reasons:

a) The de-gating would only have to occur for the reference instruction. That is, a reference instruction is replaced by control signals from the CLIW array. Looking at Fig.3, the execution unit can either receive controls from the decoder 40 (i.e., from a reference instruction) or from the CLIW array 70 (i.e., from a CLIW instruction). When a regular instruction exists, the controls are supplied by the decoder and not by the CLIW array. When a reference instruction exists, control signals are supplied by the CLIW array and not by the decoder (the reference instruction controls are replaced before they reach they execution stage). Consequently, the decoder must be configured such that control signals are not sent to the execution stage in response to a reference instruction. The de-gating will allow for this.

31. Applicant argues the novelty/rejection of claim 11 on page 12 of the remarks, in substance that:

> "Ito teaches that an instruction buffer stores instructions prior to decoding. Thus, at the very least, the instruction buffer of Ito is not an execution queue - the instructions stored in Ito's instruction buffers cannot be executed because they have not yet been decoded. One of ordinary skill in the art would understand that an execution queue stores decoded instructions."

32. These arguments are not found persuasive for the following reasons:

a) The examiner asserts that applicant is reading "execution queue" too narrowly. Any queue in a processor can be considered an execution queue as it is some way associated with the execution process. Furthermore, Ito's decoder buffer holds instructions until they are ready for dispatch. Instructions cannot be dispatched until they are decoded. While it may be true that the instructions are inserted in the queue prior to decoding, they are surely held in the queue until ready for dispatch, which means that at some point while in the queue, they are decoded.

33. Applicant argues the novelty/rejection of claim 21 on page 13 of the remarks, in substance that:

> "Lavi teaches that information that will be read from the CLIW array is fetched according to a CLIW reference instruction - the CLIW reference instruction is fetched from the same pathway as regular instructions... Lavi has not need for a branch instruction back to the normal instructions because they are fetched along the same pathway as the reference instructions."

34. These arguments are not found persuasive for the following reasons:

a) The examiner believes that applicant may be confused as to what the examiner is interpreting the second instruction form/set as being. The second instruction form is not the reference instruction but the instructions in the CLIW array. Regular instructions and CLIW instructions are not fetched form the same location. CLIW instructions come from the CLIW array while the regular instructions do not. A reference instruction will cause a switch to fetching from the CLIW array, and as the examiner stated, if a branch instruction is one of the instructions in the CLIW, which would be expected since braches are very well known instructions, then an instruction will once again be fetched from the regular pathway (i.e., switching is performed such that first form instructions are once again fetched).

Conclusion

35. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL.** See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

DJH
David J. Huisman
July 18, 2006

Eddie Chan

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100